A Few Good Predictions: Selective Node Labeling in a Social Network

Gaurish Chaudhari
IIT Bombay
gsc.chaudhari@gmail.com

Vashist Avadhanula
IIT Bombay
vas1089@gmail.com

Sunita Sarawagi
IIT Bombay
sunita@iitb.ac.in

ABSTRACT

Many social network applications face the following problem: given a network $G = (V, E)$ with labels on a small subset $\mathcal{O} \subset V$ of nodes and an optional set of features on nodes and edges, predict the labels of the remaining nodes. Much research has gone into designing learning models and inference algorithms for accurate predictions in this setting. However, a core hurdle to any prediction effort is that for many nodes there is insufficient evidence for inferring a label.

We propose that instead of focusing on the impossible task of providing high accuracy over all nodes, we should focus on *selectively* making the few node predictions which will be correct with high probability. Any selective prediction strategy will require that the scores attached to node predictions be well-calibrated. Our evaluations show that existing prediction algorithms are poorly calibrated. We propose a new method of training a graphical model using a conditional likelihood objective that provides better calibration than the existing joint likelihood objective. We augment it with a decoupled confidence model created using a novel unbiased training process. Empirical evaluation on two large social networks show that we are able to select a large number of predictions with accuracy as high as 95%, even when the best overall accuracy is only 40%.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Parameter Learning

Keywords

well calibrated probabilities; graphical models; high confidence predictions

1. INTRODUCTION

The problem of predicting labels for users in a partially labeled social network has many applications. For example, in the Twitter follower network, a recent problem [17, 9, 12, 10, 22] is predicting a user location starting from a small 1–2% known locations obtained from users who allow geo-tagging their tweets. These predictions are used to improve user experience through location based services like recommendations, advertisements, and automatic language selection. As another example consider a friendship network like Facebook, where a small subset of users provide profile attributes such as age, gender, education, and, hobbies. A problem of recent interest [28, 19, 31] is to predict the missing attributes of other users based on known attributes of their friends.

A core problem faced by all prediction algorithms is that for most users there is just not sufficient signal to infer their labels. For example, existing efforts on location prediction of Twitter users report accuracy in the range of 50–60% [17, 9, 12, 10]. This means that almost half of the predicted locations are wrong, leading to misplaced recommendations sometimes bordering on to downright absurd. Our goal is to abstain from making such prediction. Our motto is *pauca sed matura*[1] — make a few predictions but ensure that they are correct with high probability. More formally, our problem statement is as follows.

Given a social network $G = (V, E)$, known labels for a small subset $\mathcal{O} \subset V$, and a confidence threshold σ, predict the labels of the largest possible V such that the predictions are correct with at least σ probability. Equivalently, attach a *well-calibrated* probability p_u with the predicted label of each node u where well-calibrated p_u values have the property that the accuracy over nodes u with $p_u \geq \sigma$ is at least σ.

Such a selective labeling problem is well-motivated in large social networks settings because such networks exhibit very high diversity in accuracy in different regions arising out of differences in the level of connectivity of users, the degree of homophily of the predicted labels, and the fraction of known labels. A node labeling algorithm just needs to translate the diversity to correctness probability of the predicted label. Algorithms for labeling nodes in a partially labeled graph have been extensively researched across multiple communities (see [24, 21, 16, 3, 14] for a survey). Many techniques have been harnessed, including, iterative classification [5, 15, 24], Markov Random Fields (MRFs) [29, 4, 27], random walks [2, 25], label propagation [26], semi-supervised learning [6, 32], communities [19] and co-citations [3]. When we compared many of these on their ability to provide well-calibrated probabilities, we found that labeling probabilities were poorly calibrated in all methods, including MRF-based methods which create a joint distribution over labels in a

WSDM'14, February 24–28, 2014, New York, New York, USA.
Copyright 2014 ACM 978-1-4503-2351-2/14/02 ...$15.00.
http://dx.doi.org/10.1145/2556195.2556241 .

[1]*few but ripe .... Carl Friedrich Gauss*

graph. While many studies [24, 16, 18, 21] have compared methods on overall accuracy, ours is the first comparison on calibration quality.

In this paper we propose a method for labeling nodes in a partially labeled graph and attaching well-calibrated probabilities to the predicted label. Our approach is based on MRFs, chosen because of their popularity in the graph labeling community and their elegant modeling of the joint distribution over all labels in a graph. We make two key modifications to MRFs to make them better calibrated.

First, we propose a novel training objective: *maximize likelihood of each known label conditioned on every other known label.* This objective is more directly linked to our calibration goal than the conventional objective: *maximize joint likelihood of all known labels.* Also, it is both theoretically and empirically superior to another popular objective: *maximize pseudo-likelihood.* We address the computational challenge of training our objective through a theoretically sound graph pruning algorithm that guarantees an ϵ approximation on trees. Second, we propose a decoupled stage of confidence estimation based on marginals obtained from inference on the MRF. A challenge in training such two stage models is providing unbiased labeled data to both stages without hard partitioning the data across them. We propose a novel leave one out inference strategy on graphs to tackle this problem and provide an efficient algorithm for implementing the strategy.

Experiments on two large social networks show that our two-stage strategy is much better calibrated than any of the existing techniques and provides a practical solution to the selective prediction problem on graphs. On a one million node Twitter location prediction problem, with only 2% observed locations (a typical situation on the real Twitter dataset), the best overall accuracy by any graph prediction algorithm was only 30% whereas our method could select 60,000 users with accuracy over 80%. The best existing approach could select only one-fifth of this number.

The rest of the paper is organized as follows. Section 2 sets up the problem and notations. Section 3 discusses existing algorithms. Section 4 describes our approach. Theoretical results on pruning are in Section 5. Empirical comparisons appear in Section 6 and conclusions in Section 7.

2. PROBLEM SETUP AND NOTATIONS

Let $G = (V, E)$ be our social network where V is the set of n users and E the set of pairwise homophilic relationships among them. Let $\mathcal{O} \subset \{1, \ldots, n\}$ denote the set of k users each of whose label is known and let $R = V - \mathcal{O}$. Without loss of generality assume $\mathcal{O} = \{1, \ldots, k\}$. Let $\mathcal{Y}$ denote the space of possible labels, and $c_u \in \mathcal{Y}$ for each $u \in \mathcal{O}$ denote user u's label. Let $\mathbf{c}$ denote the vector $c_1, \ldots, c_k$, and m denote the cardinality $|\mathcal{Y}|$.

In general, each node and edge in G is associated with a set of features which is either user-provided (e.g. the education of a user may help predict age) or derived from properties of G (e.g, the strength of an edge may depend on the log(degree) of its endpoints). We use $\mathbf{f}(\ell, u)$ to denote the vector of node features when user u takes label ℓ, and $\mathbf{f}(\ell, \ell', u, v)$ to denote the edge feature vector on edge (u, v) when label of u is ℓ and v is ℓ'. Often the node feature vector is empty, and the edge features may only depend on whether ℓ and ℓ' are equal. We do not consider features over larger subsets of variables, for example a node and all its neighbors, because most such features can be rewritten as sum of edge features.

Our goal is to infer a label $\hat{c}_u$ for each remaining user $u \in V - \mathcal{O}$ based on G and the features, and attach a probability p_u that the prediction is correct. Typically, this task is performed in two steps. First we *train* parameters of a model using observed labels. We will generically call these parameters as $\mathbf{w}$. Second, we *deploy* the trained model to get $(\hat{c}_u, p_u)$ for each $u \in R$.

In this paper we use bold-faced symbols for vectors (e.g. $\mathbf{c}, \boldsymbol{\mu}_j$) and normal font for their members indexed either as a subscript (e.g. c_i) or as an argument (e.g. $\mu_j(i)$).

3. EXISTING/RELATED APPROACHES

We consider two types of existing solutions to this problem: iterative classification based (Section 3.1) and graphical model based (Section 3.2). Readers familiar with these approaches can directly skip to our approach in Section 4. We do not consider methods based on random walks [2, 25], label propagation [6, 26], and other pure inference algorithms [19] because these either do not support learning with both node and edge features, or are subsumed by the graphical model based approach. Another promising approach based on Gaussian Markov Models is Copula Latent Markov models [30]. These do not fit our setup because they require the labels to be binary and do not model features on edges. Our datasets span thousands of labels and heavily depend on edge features.

Algorithm 1 Framework: training and deployment

```
Input: $G = (V, E), \mathbf{f}, \mathcal{O}, \mathbf{c}$
Initialize $\mathbf{w}$.
while $\mathbf{w}$ not converged do
    E-step Call inference($G, \mathbf{w}, \mathbf{c}$) and obtain prediction $\hat{c}_u$
    and marginals $\mu_u(\ell)$ for $u \in R, \ell \in \mathcal{Y}$.
    M-step Re-estimate $\mathbf{w}$ using $(\hat{c}_u, \boldsymbol{\mu}_u)$ for each $u \in R$
    and $c_u$ for each $u \in \mathcal{O}$.
end while
Deployment Call inference($G, \mathbf{w}, \mathbf{c}$) to get $(\hat{c}_u, \mu_u(\hat{c}_u))$
for $u \in R$.
```

3.1 Iterative classification

This popular approach from relational learning [5, 15, 24, 20] consists of repeatedly creating a classifier to predict a node's label using features of the node and labels of its neighbors in the graph, only some of which may be observed. In Algorithm 1 we present a common framework to describe the many existing variants of this approach. The first step is to initialize the parameters $\mathbf{w}$ of the classifier by training on only the observed node and its observed neighbors. Next, we enter a EM kind of loop that trains the classifier $\Pr(x_u|\mathbf{c}_{N(u)}, \hat{c}_{N(u)}, \boldsymbol{\mu}_{N(u)})$ which predicts x_u conditioned on known label c_v, or predicted label $\hat{c}_v$ and/or distribution $\boldsymbol{\mu}_v$ over possible labels of each neighbor v of u. The loop consists of two steps:

E-step: First, using the existing parameters $\mathbf{w}$ infer the unknown labels $\hat{c}_u$ and marginal probability $\mu_u(\ell)$ that an unobserved node u takes label ℓ. We call this the *inference* step. Inference is also iterative and is known by several names, including ICM, relaxation labeling, and mean field inference [21].

M-step: Second, we retrain the classifier so as to maximize likelihood of the observed nodes conditioned on the (re)estimated labels/marginals of all its neighbors. This is like normal classifier training except features are aggregate over predicted/true labels/marginals of neighbors.

We compare with four variants: the ICA method that ignores the marginals $\boldsymbol{\mu}_{N(u)}$ in $\Pr(x_u|\mathbf{c}_{N(u)}, \hat{c}_{N(u)}, \boldsymbol{\mu}_{N(u)})$ and the ICA-Soft method that includes them and ignores the hard predictions $\hat{c}_{N(u)}$, the ICA-NoLoop method which after initialization, skips the training loop; and the LR method that uses only the observed nodes for both training and deployment and is thus a simple logistic regression classifier.

3.2 Graphical model based approach

The repeated classification approach does not create a consistent global distribution, and expresses variable dependence in ad hoc terms. An undirected graphical model or a Markov Random Field provides a more principled joint distribution among interacting variables in a graph and has served as another popular choice for label prediction in social networks [29, 4, 27, 24]. Let $\mathbf{x} = x_1, \ldots, x_n$ be variables denoting the labels of all n users and let $\mathbf{F}(\mathbf{x})$ denote the sum of the feature vectors over the entire graph, that is

$$\mathbf{F}(\mathbf{x}) = \sum_{i}^{n} \mathbf{f}(x_i, i) + \sum_{(i,j)\in E} \mathbf{f}(x_i, x_j, i, j) \tag{1}$$

A graphical model expresses the joint distribution as

$$\Pr(x_1, \ldots, x_n|\mathbf{w}) = \frac{1}{Z(\mathbf{w})} e^{\mathbf{w}.\mathbf{F}(x_1,\ldots,x_n)} \tag{2}$$

where $Z(\mathbf{w})$ is a normalizer called the partition function and is equal to $\sum_{x_1,\ldots,x_n} e^{\mathbf{w}.\mathbf{F}(x_1,\ldots,x_n)}$

Many different methods have been proposed to train $\mathbf{w}$. We discuss one of the most popular of these: the joint likelihood method. This method maximizes the joint probability of all observed labels as follows

$$\begin{aligned} &\max_{\mathbf{w}} \log \Pr(\mathbf{x}_{\mathcal{O}} = \mathbf{c}|\mathbf{w}) \\ &= \max_{\mathbf{w}} \log \sum_{\mathbf{x}_R} e^{\mathbf{w}.\mathbf{F}(\mathbf{c},\mathbf{x}_R)} - \log Z(\mathbf{w}) \end{aligned} \tag{3}$$

The objective is not convex when R is non-empty because of the summation within the log. Therefore, typically the EM algorithm is used for finding a local optima.

E-step: The E-step computes marginal probability of each node and edge conditioned on the observed labels. That is, for each node compute $\mu_u(\ell) = \Pr(x_u = \ell|\mathbf{x}_{\mathcal{O}} = \mathbf{c}, \mathbf{w})$ and for each edge compute $\mu_{u,v}(\ell, \ell') = \Pr(x_u = \ell, x_v = \ell'|\mathbf{x}_{\mathcal{O}} = \mathbf{c}, \mathbf{w})$ by marginalizing the joint distribution in Equation 2. These can be computed simultaneously for all nodes and edges using one of the many existing algorithms for inference in graphical models [13]. However, on general graphs, exact inference is intractable, and many approximations exist, including Gibbs Sampling, Mean-field, Belief Propagation (BP) and its convergent variants such as TRWS [13].

M-step: The M-step solves for the expected log likelihood of the observed node which becomes

$$\begin{aligned} &\mathbf{w}.E_\mu(\mathbf{F}(\mathbf{c}, \mathbf{x}_R)) - \log Z(\mathbf{w}) \\ &\text{where } \; E_\mu(\mathbf{F}(\mathbf{c}, \mathbf{x}_R)) = \\ &\sum_{(i,j)\in E,\ell,\ell'} \mu_{ij}(\ell,\ell')\mathbf{f}(\ell,\ell',i,j) + \sum_{i\in V,\ell} \mu_i(\ell)\mathbf{f}(\ell,i) \end{aligned} \tag{4}$$

This objective is convex in $\mathbf{w}$ and can be solved using gradient descent. However, a new challenge is that computation of $\log Z(\mathbf{w})$ is intractable and approximate inference via methods like BP or sampling does not guarantee a convergent gradient descent loop. Recent research [23, 11] provides an elegant solution to this problem by approximating $\log Z(\mathbf{w})$ as a convex optimization over message variables and jointly solving for the message variables and $\mathbf{w}$ as one convergent program.

After training parameters $\mathbf{w}$, a final inference step (same as in the E-step of training) is used to compute the marginals $\mu_u(\ell) = \Pr(x_u = \ell|\mathbf{x}_{\mathcal{O}} = \mathbf{c}, \mathbf{w})$ for each $u \in R, \ell$ and predict label $\hat{c}_u = \text{argmax}_\ell \mu_u(\ell), \;\; \forall u \in R$. The confidence p_u of the prediction is just $\mu_u(\hat{c}_u)$.

Since the confidence is marginal probability of a distribution trained via a statistically sound training objective, we expected the confidence values to be well-calibrated. But, when we deployed them on two real-life social networks, we did not find that to be the case at all. Therefore, in the next section we present the steps we took to solve this problem.

4. OUR APPROACH

We address the problem of getting well-calibrated probabilities through fixes on two fronts. First, we link the graphical model training objective more directly to our calibration goal (Section 4.1). Second, we decouple the confidence estimation problem from label prediction and develop a method of collecting an unbiased labeled set for training a well-calibrated estimator of confidence (Section 4.2).

4.1 Modified training of graphical models

We observed an impedance mismatch in the graphical model approach (Section 3.2) between the training objective of maximizing *joint probability of all observations* $\Pr(\mathbf{x}_{\mathcal{O}} = \mathbf{c}|\mathbf{w})$, and the predicted *single variable marginals conditioned on all observation* $\Pr(x_u = \hat{c}_u|\mathbf{x}_{\mathcal{O}} = \mathbf{c})$. We propose to fix this mismatch through a training objective that maximizes conditional likelihoods where for each observed label c_i we maximize its probability conditioned on all other observed labels. Our revised training objective then becomes:

$$\begin{aligned} &\max_{\mathbf{w}} \log \prod_{i\in\mathcal{O}} \Pr(x_i = c_i|\mathbf{x}_{\mathcal{O}-i} = \mathbf{c}_{-i}, \mathbf{w}) \\ &= \max_{\mathbf{w}} \sum_{i\in\mathcal{O}} \log \sum_{\mathbf{x}_R} e^{\mathbf{w}.\mathbf{F}(\mathbf{c},\mathbf{x}_R)} - \log \sum_{\mathbf{x}_{R\cup\{i\}}} e^{\mathbf{w}.\mathbf{F}(\mathbf{c}_{-i},\mathbf{x}_{R\cup\{i\}})} \end{aligned}$$

where the condition $\mathbf{x}_{\mathcal{O}-i} = \mathbf{c}_{-i}$ implies that we are fixing the labels of all observed nodes except the ith one. We call this the node conditional likelihood method or NCL in short. Likewise, we use the short form JL for the Joint Likelihood method. We next elaborate on the steps required for solving this objective efficiently. This objective is also non-convex but this is only due to the first term which is the same as in JL (except for being repeated k times). Therefore we use the same EM framework to solve for a local optimum.

E-step: Same as in JL.

M-step: The M-step is different because of difference in the second term. We need to solve for expected conditional likelihood of each observed node which becomes:

$$\sum_i \mathbf{w}.E_\mu(\mathbf{F}(\mathbf{c}, \mathbf{x}_R)) - \log \sum_{\mathbf{x}_{R\cup\{i\}}} e^{\mathbf{w}.\mathbf{F}(\mathbf{c}_{-i},\mathbf{x}_{R\cup\{i\}})} \tag{5}$$

where $E_\mu(\mathbf{F}(\mathbf{c}, \mathbf{x}_R))$ is same as in Equation 4. The second term looks like a partition function and we denote it as $\log Z(\mathbf{w}|\mathbf{c}_{-i})$. In order to compute $\log Z(\mathbf{w}|\mathbf{c}_{-i})$ we need to separately run inference for each observed node because the conditioning variables are different for each i. There is no easy way to reuse computation across different i values and requires inference on the full graph k times for each optimization loop. We propose an approximation to reduce this cost. For computing $\Pr(x_i = c_i|\mathbf{x}_{\mathcal{O}-i} = \mathbf{c}_{-i}, \mathbf{w})$ (henceforth denoted as $\Pr(c_i|\mathbf{c}_{-i})$) we select a mini-graph G_i from G such that the approximate probability $\Pr_{G_i}(c_i|\mathbf{c}_{-i})$ computed only using features of variables in G_i is within a user-given tolerance ϵ of $\Pr(c_i|\mathbf{c}_{-i})$. In Section 5 we present how such mini-graphs are selected. Let R_i denote the subset of R in G_i and let $\mathbf{F}_{G_i}(\mathbf{x})$ denote the sum of features over nodes and edges in G_i. The modified M-step now becomes:

$$\sum_i \mathbf{w}.E_\mu(\mathbf{F}_{G_i}(\mathbf{c}, \mathbf{x}_R)) - \log \sum_{\mathbf{x}_{R_i \cup \{i\}}} e^{\mathbf{w}.\mathbf{F}_{G_i}(\mathbf{c}_{-i}, \mathbf{x}_{R \cup \{i\}})}$$

Computationally, this objective for each i is not too different from the M-step of joint likelihood except that it is performed over a smaller graph G_i. As in the joint likelihood case, even though the objective is convex, the computation of $\log Z(\mathbf{w}|\mathbf{c}_{-i})$ is intractable on arbitrary G_is. Therefore, here also we rely on the convex approximation of [23, 11]. This provides us an efficient and convergent M-step.

After training parameters $\mathbf{w}$, we use inference to compute $(\hat{c}_u, \mu_u(\hat{c}_u))$ $\forall u \in R$ exactly as in JL.

4.1.1 *Asymptotic statistical guarantees*

We can prove that like JL, the NCL objective is also consistent. This means that if the model is faithful to the true data distribution, and if labeled data is infinite, then the NCL objective will find the true parameters. We skip a proof due to lack of space but the broad steps are the same as in Theorem 20.3 of [13]. Also, our EM-algorithm is guaranteed to find a locally optimum solution as in JL. This is easy to see because EM only modifies the first non-convex term in the NCL objective and this term is identical to the first term of JL up to a positive multiplicative constant.

4.1.2 *Relationship to other training objectives*

Some readers might find our NCL objective similar to another popular training objective [29, 13] called Pseudo-likelihood (PL). PL is based on approximating the joint probability $\Pr(x_1, \ldots, x_n)$ as $\prod_{u=1}^n \Pr(x_u|\mathbf{x}_{N_u})$, the product of node-level probability conditioned on the node's neighbors. When all neighbors are known, PL is identical to NCL since $\Pr(x_i = c_i|\mathbf{x}_{\mathcal{O}-i} = \mathbf{c}_{-i}, \mathbf{w}) = \Pr(x_i = c_i|\mathbf{x}_{N_i} = \mathbf{c}_{N_i}, \mathbf{w})$. However, for the general case with unknown neighbors, the only known extension of PL we are aware of is the PL-EM method of [29] which is very different from NCL. PL-EM proposes an approximate EM-based solution of the PL objective where in the E-step mean-field inference is used to compute the marginals $\boldsymbol{\mu}_u$ for each $u \in R$. The M-step then solves the following convex objective:

$$\sum_{i \in \mathcal{O}} \sum_{\mathbf{x}_{R_i^N}} \prod_{u \in R_i^N} \mu_u(x_u) \log \Pr(x_i|\mathbf{x}_{R_i^N}, \mathbf{x}_{\mathcal{O}_i} = \mathbf{c}_{\mathcal{O}_i}, \mathbf{w}) \qquad (6)$$

where R_i^N denotes the unobserved neighbors of node i and $\mathcal{O}_i$ denote the observed neighbors of i. Unlike our approach, PL-EM is not statistically consistent because its M-step attempts to maximize expected likelihood with respect to the *mean field distribution* which is not guaranteed to match the true distribution. We will also contrast the two approaches empirically in Section 6.

4.2 Decoupled confidence estimation

We now describe a second step that we took to better calibrate the confidence of predictions $\hat{c}_u$ from inference on the graphical model. While the marginals $\mu_u(\hat{c}_u)$ of NCL are better calibrated than of JL, our experiments show that they are still not good enough for selective node labeling. Both methods tended to provide overly inflated values of $\mu_u(\hat{c}_u)$ for immediate neighbors of observed nodes, and there was no easy way to offset that while also ensuring effective label propagation in the graph. We therefore chose to train a decoupled model $\mathcal{C}$ that works on $\hat{c}_u$ and marginals $\mu_u(.)$ produced by the graphical model and outputs a confidence p_u of the prediction $\hat{c}_u$ being correct for each $u \in R$.

Assume that we know true labels of a subset of nodes $D \subset R$ that does not overlap with $\mathcal{O}$. Then $\mathcal{C}$ can be a probabilistic binary classifier, like a logistic regression model trained using D as follows. For each $j \in D$, create an instance with a label $y_j = 1$ if $\hat{c}_j = c_j$ and $y_j = 0$ otherwise, and a set of features $\mathbf{z}_j$ derived from observed and predicted labels/marginals in j's neighborhood. Examples of such features include, the smoothed fraction of j's neighbors that have label $\hat{c}_j$, the marginal $\mu_j(\hat{c}_j)$, the largest fraction of nodes having a label other than $\hat{c}_j$ in j. The complete list of features we used can be found in [7].

A major shortcoming of the above method is that we need a labeled set D to train $\mathcal{C}$ in addition to the set $\mathcal{O}$ for prediction. Since our goal is to maximize the number of nodes correctly predicted, we want $\mathcal{O}$ to be as large as possible. Setting aside a portion of $\mathcal{O}$ as D will compromise this goal, and/or could lead to a poorly trained $\mathcal{C}$.

We propose a novel method of creating an unbiased training set for $\mathcal{C}$ without any additional labeled data. When we perform inference on the full graph to get predictions $\hat{c}_u$ on the unobserved set R, the labels of nodes in $\mathcal{O}$ are pinned to $\mathbf{c}$, and therefore get no predicted label. Our key idea is to use the mini-graphs created during NCL training, to run inference on each G_i with all but the ith observed label $\mathbf{c}_{-i}$ in G_i to obtain marginals $\bar{\mu}_u^i(\ell) = \Pr(x_u = \ell|\mathbf{c}_{-i}, \mathbf{w})$, and prediction $\bar{c}_u^i = \text{argmax}_\ell \bar{\mu}_u^i(\ell)$ for each $i \in \mathcal{O}$ and $u \in \{i\} \cup N(i)$. Note, we used a different symbol for these marginals computed without c_i to distinguish from the marginal obtained using all of $\mathbf{c}$. This step is efficient because G_i is small and in Section 5 we will show how to obtain such a G_i. Now, for each $i \in \mathcal{O}$ create a labeled instance with label $y_i = 1$ if $\bar{c}_i^i = c_i$ else $y_i = 0$, and features $\mathbf{z}_i$ created as above. The y_i labels of the observed nodes can also be used to create additional features such as the fraction of wrongly predicted observed nodes in a node's neighborhood.

Algorithm 2 presents our overall approach.

4.2.1 *Related work on confidence estimation*

The use of a decoupled binary model to get confidence with predictions is not new, for example, [4] uses one such for selecting nodes for active labeling. The novelty of our approach is in the way we create an unbiased labeled dataset for training while using the same data for predictions. In contrast, [4] assumes a separate labeled dataset. We show in Section 6.3.1 how that impacts selective accuracy.

Algorithm 2 Our approach.

```
Input: $G = (V, E), \mathcal{O}, \mathbf{c}$
$\mathbf{w}$ = Trained parameters using NCL (Section 4.1).
Call inference($G, \mathbf{w}, \mathbf{c}$) and get $(\hat{c}_u, \boldsymbol{\mu}_u)$ for $u \in R$
for $i \in \mathcal{O}$ do
    $G_i$ = Prune $G$ to approximate $\Pr(c_i|\mathbf{c}_{-i})$ (Section 5)
    Call inference($G_i, \mathbf{w}, \mathbf{c}_{-i}$), get $(\bar{c}^i_u, \bar{\boldsymbol{\mu}}^i_u)$ $u \in \{i\} \cup N(i)$
    $\mathbf{z}_i$ = Make_ Features($\bar{c}^i_i, \bar{\boldsymbol{\mu}}^i_i, \bar{\mathbf{c}}^i_{N(i)}, \bar{\boldsymbol{\mu}}^i_{N(i)}, \mathbf{c}_{N(i)}$)
    $y_i = 1$ if $\bar{c}^i_i = c_i$, $y_i = 0$ otherwise.
end for
$\mathcal{C}$ = logistic model trained using $\{(\mathbf{z}_i, y_i) : i \in \mathcal{O}\}$
for $u \in R$ do
    $\mathbf{z}_u$ = Make_ Features($\hat{c}_u, \boldsymbol{\mu}_u, \hat{c}_{N(u)}, \boldsymbol{\mu}_{N(u)}, \mathbf{c}_{N(u)}$)
    $p_u = \mathcal{C}(\mathbf{z}_u)$
end for
return $\{(u, p_u) : u \in R\}$
```

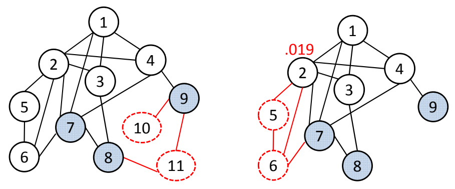

Figure 1: An example of graph pruning to obtain $\Pr(c_1|\mathbf{c}_{-1})$. The shaded nodes are observed. The dotted nodes in the left are nodes pruned via the Markov condition. The dotted nodes in the right are nodes pruned using $\epsilon = 0.02$ via the method of Section 5.3.

5. PRUNING THE GRAPHICAL MODEL

We show how to select a sub-graph G_i of G such that the approximate probability $\Pr_{G_i}(c_i|\mathbf{c}_{-i})$ computed only using G_i is within ϵ of $\Pr(c_i|\mathbf{c}_{-i})$. First, based on the Markov property of graphical models, we can prune any node u from G all of whose paths to i are through observed nodes without modifying $\Pr(c_i|\mathbf{c}_{-i})$. For example, in Figure 1, we can prune nodes 10 and 11 when $i = 1$ because for both of them the only paths to node 1 are through observed nodes (shown shaded). Call the pruned graph G_i. G_i might still be large and we present recipes for pruning further nodes from it while maintaining an ϵ approximation. For ease of notation, we will drop the subscript i, use G instead of G_i, assume $i = 1$, $c_i = 1$, and assume that the nodes in G are numbered $1, \ldots, n$. Also, we denote the node potentials as $\psi_u(x_u)$ and define $\psi_u(x_u) = e^{\mathbf{w}\cdot\mathbf{f}(x_u,u)}$ for $u \in (R \cap G_i) \cup \{i\}$, $\psi_u(x_u) = M$ if $u \in (\mathcal{O} - i) \cap G_i, x_u = c_u$ and $\psi_u(x_u) = 1$ otherwise where $M \gg 1$ is a large constant. We denote edge potentials as $\psi_{uv}(x_u, x_v) = e^{\mathbf{w}\cdot\mathbf{f}(x_u,x_v,u,v)}$. We assume that edge features $\mathbf{f}(x_u, x_v, u, v)$ depend only on whether $x_u = x_v$. Therefore, they can be expressed as $\psi_{uv}(x_u, x_v) = (\alpha_{uv}$ if $x_u = x_v, 1$ else) where α_{uv} is a constant. Further, since we assume homophily, $\alpha_{uv} \geq 1$. Let α be an upper bound on the values of α_{uv}. During parameter training when the potentials are unknown, we assume that the user can guess a suitably tight bound for α as pruning is performed in terms of α. This is not too difficult in our experience, particularly since the penalty for over-estimate is just reduced pruning.

With these notations we can cast the conditional probability $\Pr(x_1 = 1|\mathbf{x}_{\mathcal{O}-1} = \mathbf{c}_{-1})$ as this marginal probability

$$\Pr(x_1^1) = \frac{\sum_{x_2,\ldots,x_n:x_1=1}\prod_u \psi_u(x_u)\prod_{(u,v)}\psi_{uv}(x_u,x_v)}{\sum_{x_1,x_2,\ldots,x_n}\prod_u \psi_u(x_u)\prod_{(u,v)}\psi_{uv}(x_u,x_v)}$$

where we use the shorthand x_1^1 for $x_1 = 1$. We introduce a convenient normalization operator $\mathcal{N}(\mathbf{z})$ that when applied on a vector $\mathbf{z}$ normalizes its entries so they sum to one. Using this, we can write

$$\Pr(x_1 = \ell) = \mathcal{N}(\sum_{x_2,\ldots,x_n}\prod_u \psi_u(x_u)\prod_{(u,v)}\psi_{uv}(x_u,x_v))(\ell)$$

Our goal is to remove nodes from G so as to approximate $\Pr(x_1^1)$ within a given ϵ of the unpruned value. We develop the method in three stages: first assume that G is a chain with x_1 at one end, then generalize to the case when G is a tree, and finally to arbitrary graphs.

The proofs in the rest of the section are quite technical and assume knowledge of message-based computations in graphical models. Readers unfamiliar with the topic can skip the proofs and just see the main pruning results in Theorems 5.1, 5.2, and 5.3.

5.1 Single chain

Assume the chain of nodes is $x_1, \ldots, x_n$ with an edge between each x_u and x_{u+1}. In a chain if we remove a node x_t, all nodes x_j for $j > t$ are also removed. Let $P(x_1|\psi_{1\ldots t})$ denote the marginal calculated with potentials up to node t. Then $P(x_1|\psi_{1\ldots t}) =$

$$\mathcal{N}\left(\psi_1(x_1)\sum_{x_2,\ldots,x_t}\prod_{u=2}^{t}\psi_u(x_u)\psi_{u-1,u}(x_{u-1},x_u)\right) \tag{7}$$

Thus, our task reduces to identifying an index t such that $|P(x_1^1|\psi_{1\ldots n}) - P(x_1^1|\psi_{1\ldots t})| \leq \epsilon$. For a chain, the impact of all potentials on node 1 after t can be expressed as a simplex message, say $\boldsymbol{\theta}_t$ that node $t+1$ sends to t and we can rewrite $P(x_1^1|\psi_{1\ldots n}) = P(x_1^1|\psi_{1\ldots t}, \boldsymbol{\theta}_t)$. The principle we follow for pruning is to find the maximum swing in this value over all possible values of $\boldsymbol{\theta}_t \in \Delta^m$, the m dimensional simplex. Let $\hat{\boldsymbol{\theta}}$ and $\check{\boldsymbol{\theta}}$ denote the two extreme values of $\boldsymbol{\theta}_t$ that maximize this difference, that is, $(\hat{\boldsymbol{\theta}}, \check{\boldsymbol{\theta}}) =$

$$\text{argmax}_{\hat{\boldsymbol{\theta}}',\check{\boldsymbol{\theta}}'\in\Delta^m}\left|P(x_1^1|\psi_{1\ldots t},\hat{\boldsymbol{\theta}}') - P(x_1^1|\psi_{1\ldots t},\check{\boldsymbol{\theta}}')\right|$$

Now, we will derive what $(\hat{\boldsymbol{\theta}}, \check{\boldsymbol{\theta}})$ should be and for what t will the difference be $\leq \epsilon$. We first introduce some notations. Let $\hat{\mathbf{M}}_r^t$ denote a message at distance r from node 1, when a message $\hat{\mathbf{M}}^t = \hat{\boldsymbol{\theta}}$ is injected from node $t+1$ to t. Likewise define $\check{\mathbf{M}}_r^t$ with message $\check{\boldsymbol{\theta}}$ from $t+1$ and $\bar{\mathbf{M}}_r^t$ with message $\boldsymbol{\theta}_t$ from $t+1$. Let $h(\boldsymbol{\psi}, \alpha, \mathbf{M})$ denote the outgoing message at a node with node potential $\boldsymbol{\psi}$ when a message $\mathbf{M}$ is injected into it via an edge with Potts potential α. That is,

$$h(\boldsymbol{\psi}, \alpha, \mathbf{M}) = \mathcal{N}(\boldsymbol{\psi}\boldsymbol{\gamma}) \text{ where } \boldsymbol{\gamma} = (\alpha - 1)\mathbf{M} + \mathbf{1}_m, \tag{8}$$

where $\mathbf{1}_m$ is a length m vector of all ones. Using this we can express $\bar{\mathbf{M}}_r^t$, recursively as follows:

$$\bar{\mathbf{M}}_r^t = h(\boldsymbol{\psi}_r, \alpha, \bar{\mathbf{M}}_{r+1}^t), \text{ if } r < |t|, \quad \bar{\mathbf{M}}_t^t = \boldsymbol{\theta}_t, \tag{9}$$

and similarly $\check{\mathbf{M}}^t, \hat{\mathbf{M}}^t$. From the above, it is easy to see that $P(x_1^1|\psi_{1\ldots t}, \boldsymbol{\theta}_t) = \bar{M}_1^t(1)$.

First, we assume no node potentials and derive a closed form expression for $\Pr(x_1 = \ell|\psi_{1\ldots t}, \boldsymbol{\theta}_t)$.

Lemma 5.1 If G is a chain, $\alpha_{uv} = \alpha$, $\psi_u(\ell) = 1\ \forall u, \ell$, then

$$\Pr(x_1 = \ell | \psi_{1\ldots t}, \boldsymbol{\theta}_t) = \frac{\theta_t(\ell) - 1/m}{(\frac{m}{\alpha-1} + 1)^{t-1}} + \frac{1}{m} \tag{10}$$

Proof. When node potential is 1 for all labels, Equations 8 and 9 gives that $\bar{M}^t_{r-1}(\ell) = \frac{(\alpha-1)\bar{M}^t_r(\ell)+1}{m+\alpha-1}$. To get $\bar{M}^t_1(\ell)$ which is equal to $\Pr(x_1 = \ell | \psi_{1\ldots t}, \boldsymbol{\theta}_t)$ we repeatedly apply this formula t times. After simplifying the result, we get the RHS.

A corollary from the above that we will find useful is:

Corollary 5.1 If $\alpha_{uv} \leq \alpha \quad \forall(u,v) \in E$ (other conditions same as in Lemma 5.1), then

$$\Pr(x_1 = \ell | \psi_{1\ldots t}, \boldsymbol{\theta}_t) - \left[\frac{\theta_t(\ell) - 1/m}{(\frac{m}{\alpha-1} + 1)^{t-1}} + \frac{1}{m}\right] \leq 0 \ \text{ if } \theta_t(\ell) \geq \frac{1}{m}$$
$$\geq 0 \ \text{ otherwise.}$$

Theorem 5.1 If G is a chain, $t > \frac{\log\frac{1}{\epsilon}}{\log(\frac{m}{\alpha-1}+1)}$, $\alpha_{uv} \leq \alpha$ $\psi_u(\ell) = 1 \quad \forall u, \ell$ then $|P(x_1^1|\psi_{1\ldots n}) - P(x_1^1|\psi_{1\ldots t})| \leq \epsilon$. In other words, pruning potentials $\psi_{t,t+1}\ldots,\psi_{n-1,n}$ changes $\Pr(x_1^1)$ by at most ϵ.

Proof.

$$\begin{aligned}
&|P(x_1^1|\psi_{1\ldots n}) - P(x_1^1|\psi_{1\ldots t})| \\
&\leq \max_{\hat{\boldsymbol{\theta}},\check{\boldsymbol{\theta}} \in \Delta^m} \left|P(x_1^1|\psi_{1\ldots t}, \hat{\boldsymbol{\theta}}) - P(x_1^1|\psi_{1\ldots t}, \check{\boldsymbol{\theta}})\right| \\
&\leq \max_{0 \leq \check{\theta}(1) \leq 1/m \leq \hat{\theta}(1) \leq 1} \left|\frac{\hat{\theta}(1) - \check{\theta}(1)}{(\frac{m}{\alpha-1}+1)^{t-1}}\right| \quad \text{(Corollary 5.1)} \\
&\leq \frac{1}{(\frac{m}{\alpha-1}+1)^{t-1}} \leq \epsilon
\end{aligned}$$

Now let us extend the result to include node potentials. Recall that since training is not over, we do not know the value of the node potentials. Unlike edge potentials, it is difficult to bound their values and without those we cannot obtain a closed form expression for $\Pr(x_1)$ in terms of $\boldsymbol{\theta}_t$ unlike what we did earlier. So, we take an alternative approach and bound the difference between the upper and lower bound of label 1 in the message at stage r $\hat{M}^t_r(1) - \check{M}^t_r(1)$ in terms of the corresponding difference $\hat{M}^t_{r+1}(1) - \check{M}^t_{r+1}(1)$ at stage $r+1$. The following lemma gives the bound.

Lemma 5.2 $\hat{M}^t_r(1) - \check{M}^t_r(1) \leq \kappa(m,\alpha)(\hat{M}^t_{r+1}(1) - \check{M}^t_{r+1}(1))$ where

$$\kappa(m,\alpha) = \frac{(\alpha-1)(m+\alpha-1)}{\alpha(m+\alpha-1)+3m+\alpha-5} \tag{11}$$

Proof. The proof is involved and we provide a brief sketch. Let $\boldsymbol{\psi}_r$ denote the unknown node potential at r. Since,

$$\begin{aligned}
\hat{M}^t_r(1) - \check{M}^t_r(1) &= h(\boldsymbol{\psi}_r, \alpha, \hat{\mathbf{M}}^t_{r+1})(1) - h(\boldsymbol{\psi}_r, \alpha, \check{\mathbf{M}}^t_{r+1})(1) \\
&\leq \max_{\boldsymbol{\psi}} h(\boldsymbol{\psi}, \alpha, \hat{\mathbf{M}}^t_{r+1})(1) - h(\boldsymbol{\psi}, \alpha, \check{\mathbf{M}}^t_{r+1})(1)
\end{aligned}$$

We prove in [1] that the solution $\boldsymbol{\psi}^*$ for the optimization problem above is

$$\psi^*(1) = \frac{\sqrt{(1-\hat{\gamma}_1)(1-\check{\gamma}_1)}}{\sqrt{(1-\hat{\gamma}_1)(1-\check{\gamma}_1)} + (m-1)\sqrt{\hat{\gamma}_1\check{\gamma}_1}}, \ \psi^*(l>1) = \frac{1-\boldsymbol{\psi}^*(1)}{m-1}$$

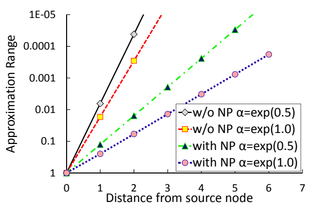

Figure 2: Error bounds (log scale) vs distance t for $m = 100$

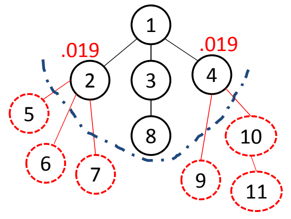

Figure 3: An example tree. Pruned nodes are dotted: $\epsilon = 0.05, m = 100, \alpha = e^{0.5}$.

where $\hat{\gamma}_1 = \frac{(\alpha-1)\hat{M}^t_{r+1}(1)+1}{m+\alpha-1}$ and likewise $\check{\gamma}_1$. Plugging in this solution we can show that:

$$\max_{\boldsymbol{\psi}} h(\boldsymbol{\psi}, \alpha, \hat{\mathbf{M}}^t_{r+1})(1) - h(\boldsymbol{\psi}, \alpha, \check{\mathbf{M}}^t_{r+1})(1) =$$
$$= \frac{\hat{\gamma}_1 - \check{\gamma}_1}{2\sqrt{(1-\hat{\gamma}_1)(1-\check{\gamma}_1)\hat{\gamma}_1\check{\gamma}_1} + \hat{\gamma}_1(1-\check{\gamma}_1) + \check{\gamma}_1(1-\hat{\gamma}_1)}$$

Next, we use a set of inequalities to show that this expression can be upper bounded by $\kappa(m,\alpha)(\hat{M}^t_{r+1}(1) - \check{M}^t_{r+1}(1))$. The details of this can be found in [1].

Theorem 5.2 If G is a chain where $\alpha_{uv} \leq \alpha$ and $t > \frac{\log\epsilon}{\log(\kappa(m,\alpha))} + 1$, then $|P(x_1^1|\psi_{1\ldots n}) - P(x_1^1|\psi_{1\ldots t})| \leq \epsilon$.

Proof. The proof follows by repeatedly applying Lemma 5.2 $t-1$ times until we get that $|P(x_1^1|\psi_{1\ldots t}, \hat{\boldsymbol{\theta}}) - P(x_1^1|\psi_{1\ldots t}, \check{\boldsymbol{\theta}})| = \hat{\mathbf{M}}^t_1(1) - \check{\mathbf{M}}^t_1(1) \leq \max_{\hat{\boldsymbol{\theta}},\check{\boldsymbol{\theta}}} \kappa(m,\alpha)^{t-1}(\hat{\theta}(1) - \check{\theta}(1)) = \kappa(m,\alpha)^{t-1}$.

In Figure 2 we plot the worst case bound on $|\Pr(x_1^1|\psi_{1\ldots n}) - \Pr(x_1^1|\psi_{1\ldots t})|$ on a chain when we prune nodes after distance t from node 1 for $\alpha = e^{0.5}, m = 100$ and $\alpha = e$ and for the two cases: without node potentials (Theorem 5.1) and with node potentials (Theorem 5.2). The bounds are tighter without node potentials but in all cases we achieve good approximation within a small t. With $\alpha = e^{0.5}$, a typical value in our datasets, we approximate to within 0.0001 and 0.019 respectively at $t = 2$.

5.2 Tree

Now consider the case where G is a tree rooted at x_1. Pruning now amounts to choosing a frontier of G so that the size of the pruned tree is smallest while ensuring that the maximum swings in the potentials outside the frontier do not impact $\Pr(x_1^1)$ by more than ϵ.

Suppose we decide to prune all nodes below a node x_t with parent p. The contribution of all potentials under x_t (inclusive of its own node potential) in computing $\Pr(x_1^1)$ can be expressed as a message $\mathbf{M}_t$ from t to p. We compute what happens to $\mathbf{M}_t$ along the path $\mathcal{P}(t)$ to x_1. Unlike for chains, the path has incoming messages from other children of intermediate nodes. For example, in Figure 3 $\mathbf{M}_9$ has to multiplied with when the message from node 10 passes through node 4 to node 1. During message passing, all incoming messages at a node u are multiplied with the node potential at u, so the messages can be treated as a modified node potential. Our bounds in the previous section assumed the most adverse node potentials. These same bounds can be used unchanged on the path ignoring all other branches on the path, we only have to modify Equation 9 to work with non-continuous indices along a path $\mathcal{P}(t)$. We use notation $\hat{\mathbf{M}}^{\mathcal{P}(t)}$ instead of $\hat{\mathbf{M}}^t$ therefore.

Thus, we know how to compute message bounds for single path in the tree. If we prune below a frontier of multiple nodes, we can simply add the bounds for each frontier node because our per-path bounds are with respect to the worst effect of messages from the rest of the tree. The final theorem for pruning a tree based on frontier of nodes appears below (We skip a formal proof due to lack of space).

Theorem 5.3 Let G be a tree rooted at 1 with potentials as per Theorem 5.2. If G_T is a subset of G obtained by removing all nodes below a frontier $T = t_1, \ldots, t_f$ such that $\sum_j \kappa(m, \alpha)^{|\mathcal{P}(t_j)|} \leq \epsilon$ then $|P(x_1^1|\psi_G) - P(x_1^1|\psi_{G_T})| \leq \epsilon$.

An example of a pruned tree obtained by using $\alpha = e^{0.5}, m = 100$ and $\epsilon = 0.05$ is shown in Figure 3. The weights on a node t is $\kappa(m, \alpha)^{|\mathcal{P}(t)|}$ and the frontier is $T = 2, 4$.

5.3 Arbitrary graph

For arbitrary graphs, inference is intractable and it is difficult to efficiently bound the influence that a message from some node u has on node 1 when there are multiple paths between them. Recently, [8] addressed this problem by choosing the shortest path between two nodes to bound this influence. We follow the same strategy. Let $\mathcal{SP}(t)$ denote the shortest path from node t to 1. Define the frontier nodes of a pruned graph $G' \subset G$ as the set of unobserved nodes in G' with at least one pruned neighbor. We choose G' with frontier nodes $T = t_1, \ldots, t_f$ such that $\sum_j \kappa(m, \alpha)^{|\mathcal{SP}(t_j)|} \leq \epsilon$. For example, the right graph in Figure 1 is obtained by pruning with $\epsilon = 0.02$. Node 2 is the only frontier node here; node 7 is not a frontier node because it is observed.

Unlike for trees, we have no approximation guarantees on $\Pr(x_1^1)$ with this pruning but since inference algorithms on general graphs are also not exact, we cannot hope to do much better. Empirically, we will show in Section 6.3.3 that our pruning method is effective.

6. EXPERIMENTS

We present an evaluation of different methods on their ability to do selective node prediction and an analysis of our method along different dimensions.

6.1 Social Networks

Our experiments were performed on two large real-life social networks, Twitter and Pokec, each with more than one million nodes. We summarize their key statistics in Table 1.
Twitter: We crawled 82 million geo-tagged tweets from all over the world over four weeks in June and July 2012. We extracted from these 3.5 million geo-tagged users and assigned to each his most frequent tweet location. We then crawled for followers of users. We removed users with more than 1000 neighbors because these are typically celebrity or media accounts, and not useful for location prediction. This left us with 3 million users. The raw user locations were in (latitude, longitude) format. Since our focus is discrete prediction, we mapped these to one of 2,113 top-populated world cities using Google's Geo-coding API[2]. Users outside these 2113 cities were removed giving rise to our final graph of 1.07 million users and 3.86 million edges.

[2]developers.google.com/maps/documentation/geocoding

	Twitter	**Pokec**
# Nodes	1,071,254	1,136,049
# Edges	3,863,698	10,773,722
% Uni-directional edges	42.87	61.24
# Labels(m)	2,113	10

Table 1: Twitter and Pokec graph statistics

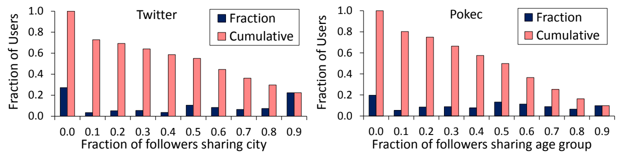

Figure 4: Histograms of fraction of followers of a user sharing his label for Twitter(left) and Pokec(right). Blue (darker) bars are for histogram and red (lighter) bars for cumulative probabilities.

Pokec: As a representative of the attribute prediction problem in a friendship network, we used the publicly[3] available Pokec social network (crawled in May 2012), a popular social network in Slovakia. Each user has a profile in Slovak spanning attributes like gender, age, hobbies, marital status, children, profession, and education. Our task is to predict a user's age, available in only 68% user profiles. We retained these 68% users, giving rise to a graph of 1.13 million nodes and 10.8 million edges. We segmented age into bins of five years, obtaining 10 age groups.

6.2 Setup and Methods

We compared eight methods: four methods from the iterative classification family (Section 3.1): LR, ICA-NoLoop, ICA, and ICA-Soft; three methods based on graphical models: JL trained on joint likelihood (Section 3.2) and PL-EM trained on pseudo likelihood (Section 4.1.2), and NCL trained on node conditional likelihood (Section 4.1); and finally our two stage method NCL+Conf that uses the decoupled training of Section 4.2. The base classifier for LR, ICA-NoLoop, ICA, and ICA-Soft was a multi-class linear logistics regression model. Methods JL and NCL used BP during deployment and convergent TRWS during training.

For both our datasets an edge could be bi-directional or uni-directional. We designed a set of nine edge features based on intuitive clues such as edges between high degree nodes indicate less homophily, bi-directional edges are stronger, and high entropy of observed labels implies less homophily. These are summarized in Table 2 and fire only for the case when $x_u = x_v$. Thus all our edge potentials are Potts. For Pokec we experimented with node features derived from a user's profile attributes like education and marital status which seemed indicative of age. However, we obtained no gain in accuracy from them. One reason could be that these were both highly unstructured text fields. Our best effort at picking high signal phrases provided little gain. In contrast, we found that our edges were highly homophilic. We ascertain this via Histograms in Figure 4 where X-axis is the fraction s_u of a user's neighbors that have the same label, bucketed into steps of 0.1, and Y-axis is the proportion of users with s_u fraction co-labeled friends. On Twitter, for more than 55% of the users, majority of their graph neighbors share his label; and for Pokec it is 51%. The edge

[3]http://snap.stanford.edu/data/soc-pokec.html

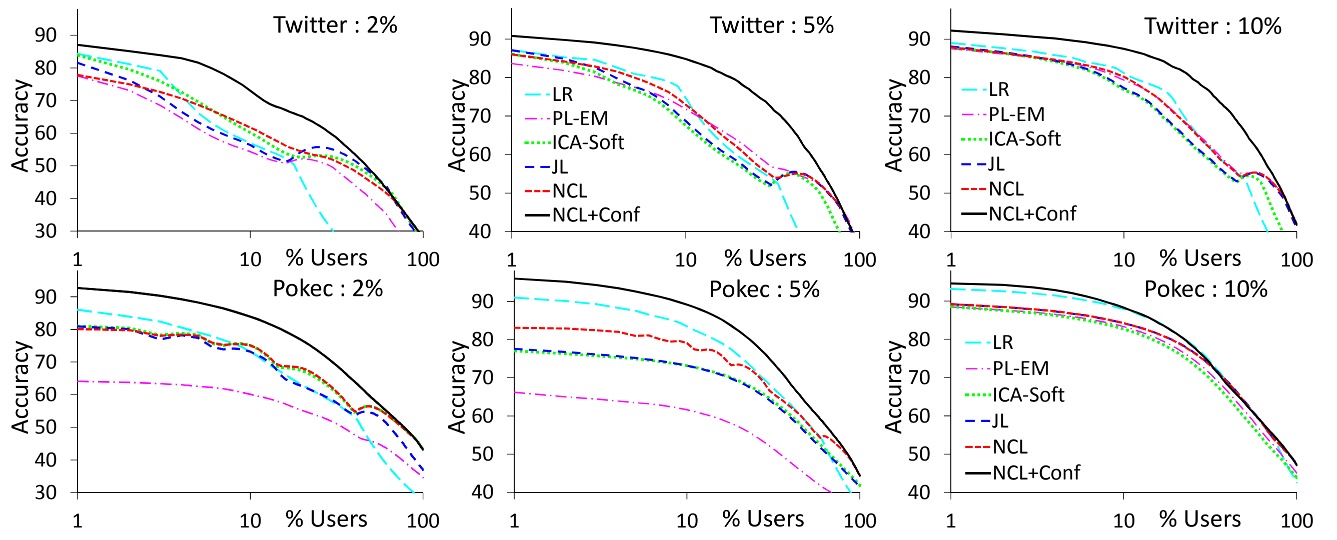

Figure 5: Precision-recall curves comparing different methods on Twitter(up) and Pokec(below) for $k = 2\%$ (left), 5% (middle), 10% (right). The legend is the same for all charts and dropped in some charts to reduce clutter. [Best seen in color]

$f_0(u,v) =$	1 (This is a bias feature)
$f_1(u,v) =$	1 if edge (u,v) is bi-directional
$f_2(u,v) =$	$\frac{1}{\log(\#\text{followers}(u)+1)}$ if u has a follower v
$f_3(u,v) =$	$\frac{1}{\log(\#\text{followees}(v)+1)}$ if u follows v
$f_4(u,v) =$	$\frac{1}{\log(\text{degree}(u)+1)} + \frac{1}{\log(\text{degree}(v)+1)}$
$f_5(u,v) =$	$\frac{1}{\log(\#\text{followers}(u)+1)} + \frac{1}{\log(\#\text{followers}(v)+1)}$ ... if u and v are follower-followee
$f_6(u,v) =$	$\frac{1}{\log(\#\text{followees}(u)+1)} + \frac{1}{\log(\#\text{followees}(v)+1)}$ ... if u and v are follower-followee
$f_7(u,v) =$	1 if u or v is observed node
$f_8(u,v) =$	2 - $\frac{\text{entropy}(\ell^O_{N(u)})-\text{entropy}(\ell^O_{N(v)})}{\log m}$ where $\ell^O_{N(i)}$ are labels of i's observed neighbors

Table 2: Edge features used by our models. The label argument is missing because they only fire when u, v get the same label.

affinity values as defined in [19] for the two datasets are 7.7 and 2.8 respectively, which compares favorably to affinities in other social networks [19]. Thus, both our datasets were trained on the nine edge features alone. The parameter vector $\mathbf{w}$ represents weight of these nine features trained via different methods. For methods from the iterative classifier family, the features at a node u were a sum of features in Table 2 over all neighbors v of u. Thus, all our methods used the same set of features and were trained with L2 regularization.

We select various subsets $\mathcal{O}$ of observed nodes in a clustered manner to simulate homophily in the level of privacy that a user prefers. Starting from a random node, we select its neighbor with probability β and teleport to some other node with probability 1-β. We set β as 0.85. We found that relative performance was not too sensitive to β and even for β as small as 0.5 we got similar curves. To select a set $\mathcal{O}$ of size k, we take k steps in this random walk. We used k values for 2%, 5% and 10% of the graph size. For each k, we select 10 random $\mathcal{O}$ sets and average over these 10 runs. We measure statistical significance using a student's t-test over the 10 seeds and accuracy at the top-p% most confident predictions for p=1,2,...,10%. Since our focus is accuracy at the top, we do not consider larger p.

6.3 Results

In Figure 5 we plot the PR curves for six[4] methods for Twitter on the first row and Pokec on the second and for $k =$ 2% on the left, $k =$ 5% in the middle, and $k =$ 10% on the right. The Y-axis is the accuracy on the top-X% predictions sorted on the confidence p_u output by each method. We used a log scale on the X-axis because our focus is accuracy on the top-few most confident predictions. The overall accuracy of a method is the Y-axis value when X-value is 100%. We make the following observations from these plots.

1. The overall accuracy of different methods ranges from 26%–41% for Twitter and 27–47% for Pokec. The variation in the overall accuracy of different methods is not of consequence because even the maximum is too low to be useful in practice.
2. If we focus on accuracy among the top 1% to 20% predictions we get a more positive story. Many methods provide high accuracy in this selected set, which touches 95% with the top 1–3% predictions. As we compare graphs from left to right for increasing k, we see that the top accuracy increases with k.
3. The best selective accuracy is provided by our decoupled approach NCL+Conf. In all cases, NCL+Conf dominates all other methods for top-20% predictions even though its overall accuracy is the same. For example, with 5% observed nodes if our goal is 85% accuracy we can find *four* and *2.1* times more predictions than any other method on Twitter and Pokec respectively. The gain of NCL+Conf is statistically significant with p-value $< 10^{-27}$ for all six cases.
4. Between the two graphical model approaches, NCL provides better calibration than JL, even though the

[4]The ICA-NoLoop and ICA methods were superseded by ICA-Soft and LR and were not plotted to reduce clutter.

$\mathcal{O}$ %	NCL+Conf	NCL	ICA-Soft	LR
Twitter				
2	**0.0/6.0**	0.0/0.6	0.0/1.4	0.0/1.2
5	**1.5/18.1**	0.0/5.1	0.08/3.5	0.15/4.6
10	**4.3/24.8**	0.0/10.2	0.2/7.7	0.3/10.1
Pokec				
2	**3.5/15.3**	0.0/0.1	0.0/2.6	0.0/4.6
5	**8.8/23.1**	0.0/7.0	0.0/0.03	2.1/14.4
10	**8.0/23**	0.02/18.7	0.02/14.4	6.8/22.6

Table 3: % of nodes selected at accuracy 90% (above the slash) & at accuracy 80% (below the slash) for Twitter and Pokec.

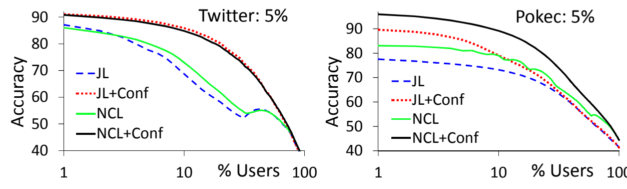

Figure 6: PR curves to compare gain of second stage on JL and NCL on Twitter and Pokec datasets with $k = 5\%$.

overall accuracies are comparable. See for example, Pokec 5%. This confirms that conditional likelihood is a better objective for selective prediction. The gain of NCL over JL is statistically significant with p-value $< 10^{-4}$ in all cases except Pokec-10% where it is 0.04.

5. The method closest to NCL from the existing literature, PL-EM, does not provide good calibration for $k = 2, 5\%$ but is okay for larger k, corroborating the conclusion of [29] that PL-EM is good with many observed nodes. On overall accuracy PL-EM is competive with JL as observed in [29], but on the top-10% accuracy, JL scores over PL-EM in all cases.
6. Among the iterative classification methods, ICA-Soft that conditions on marginal probability provides steadily better results than ICA that conditions on hard predictions. ICA-Soft is comparable to JL — slightly better than it for 2% but worse for 5% and 10%. ICA-Soft is worse than NCL in most cases.
7. The LR method is good in a narrow range but degrades rapidly, particularly for 2% observations. This is because for the few nodes with many observed neighbors the classifier is able to assign high confidence. But, there is no generalization beyond this narrow set.

As further evidence of the effectiveness of NCL+Conf for selective prediction, in Table 3 we show the fraction of nodes selected at two target accuracy values (80% and 90%) via four methods for 2, 5, and 10% observed nodes. We see that NCL+Conf is able to make selective predictions on significantly more nodes than others, particularly when size of $\mathcal{O}$ is small. For example, on Pokec 2% when the target accuracy is 90%, we are able to select 3.5% nodes whereas none of the other approach can select any node. For a target accuracy of 80%, NCL+Conf can select between $2k$ and $7k$ nodes where k is the size of $\mathcal{O}$.

6.3.1 Efficacy of our confidence estimator

Since our main gain is from the second stage confidence model, we perform two experiments to analyze this stage further. First, we check whether the lift in top-accuracy of NCL due to the second stage, holds for other methods too, e.g. JL. In Figure 6 we show PR curves for JL and JL+Conf for Twitter and Pokec with $k = 5\%$. We observe that JL+Conf is indeed much better than JL although on Pokec it continues to be worse than NCL+Conf because on that dataset even JL is much worse than NCL. Second, we check the gain due to our specific method of creating an unbiased training set over *all* observed nodes through our novel mini-graph inference method. We compare with the standard practice of splitting the observed nodes over the two stages. In Figure 7 we measure accuracy of NCL+Conf with NCL+Split which uses 70% of $\mathcal{O}$ as observed nodes for NCL inference and 30% for training the second stage. We observe that NCL+Conf shows a statistically significant gain over NCL+Split for both $k = 2\%$ and 5%.

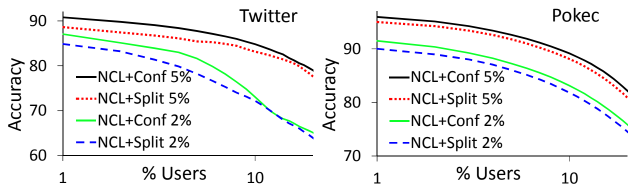

Figure 7: Comparing two different methods of training the second stage model: (1) our leave one out strategy (NCL+Conf) and (2) 70-30 split of labeled data across the two stages (NCL+Split) for k=5% and 2%.

Method	2% Twitter		5% Twitter		5% Pokec	
	Learn	*Infer*	*Learn*	*Infer*	*Learn*	*Infer*
LR	0.1	42.9	0.2	48.4	0.2	42.1
ICA-Soft	601	3971	19	216	163	14717
PL-EM	16	1265	3416	4380	4533	13115
JL	227	2378	69	2749	198	14621
NCL	846	4584	696	4994	5334	909
NCL+Conf	1414	4627	2701	5042	11771	950

Table 4: Learning and inference time in seconds.

6.3.2 Running time

In Table 4 we show the time taken for learning and inference using various methods on a sequential program on a 2.5 GHz Intel Xeon linux server with 8 GB RAM. As expected, the plain LR model is the fastest because it only looks at the small number of observed nodes during training. The running time for all graphical model-based methods show a lot of variability because the convergence of inference is dependent on the potentials, the observed nodes, and edge density. Roughly, learning+inference on Twitter takes an hour, whereas on Pokec it is three hours. The training time for NCL+Conf is two to four times more than NCL and most of it goes in computing the mini-graph instances. However, this part of the code is trivially parallelizable.

6.3.3 Pruning efficiency

In this section we analyze the running time versus accuracy trade offs due to our pruning strategy. In Figure 8 we plot average error in the probability of the true label and average inference time (log-scale) over hundred nodes for increasing sizes of the mini-graph. The X-axis varies from a minigraph size of 5 nodes to the full graph. We observe that

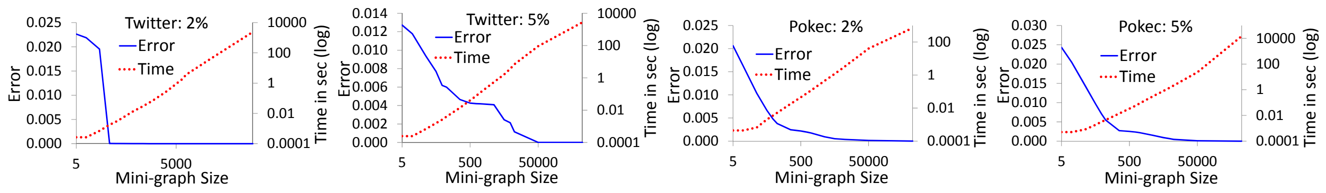

Figure 8: Error and inference time for increasingly pruned graph for Twitter and Pokec with 2% and 5% observed nodes.

the running time drops by several orders of magnitude when we prune mini-graphs so that the error is ≤ 0.002. This shows that our pruning strategy is effective for the leave one out inference needed for well-calibrated probabilities.

7. CONCLUSION AND FUTURE WORK

In this paper we defined and motivated the problem of selective prediction of labels in a social network. Our study over two large social networks revealed that existing graph-based prediction algorithms do not provide well-calibrated probabilities — a pre-requisite for selective prediction. We proposed a new node conditional likelihood objective for training a graphical model-based prediction algorithm, and a decoupled model for confidence estimation based on a novel unbiased training process. We provided theoretically sound graph pruning strategies for training the new objective. These ideas together provide an accurate and practical mechanism for selective predictions in social networks.

An important outcome of our work is that it raises even graver privacy concerns, than the ones raised in earlier studies based on overall accuracy [31]. An interesting area of future work is understanding the relationship between information blurring models and selective prediction models.

8. REFERENCES

[1] V. Avadhanula. Selective node labeling in social networks. Master's thesis, IIT Bombay, 2013.

[2] A. Azran. The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In *ICML*, 2007.

[3] S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. In *SNDA*. 2011.

[4] M. Bilgic and L. Getoor. Effective label acquisition for collective classification. In *Proc. ACM SIGKDD*, 2008.

[5] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. *SIGMOD Rec.*, 27(2):307–318, 1998.

[6] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning.* MIT Press, 2006.

[7] G. Chaudhari. High confidence predictions in social networks. Master's thesis, IIT Bombay, 2013.

[8] A. Chechetka and C. Guestrin. Focused belief propagation for query-specific inference. *JMLR*, 9:89–96, 2010.

[9] Z. Cheng, J. Caverlee, and K. Lee. You are where you tweet: a content-based approach to geo-locating twitter users. In *CIKM*, 2010.

[10] J. Eisenstein, B. O'Connor, N. A. Smith, and E. P. Xing. A latent variable model for geographic lexical variation. In *Proc. EMNLP*, 2010.

[11] T. Hazan and R. Urtasun. A primal-dual message-passing algorithm for approximated large scale structured prediction. In *NIPS*, 2010.

[12] L. Hong, A. Ahmed, S. Gurumurthy, A. J. Smola, and K. Tsioutsiouliklis. Discovering geographical topics in the twitter stream. In *Proc WWW*, 2012.

[13] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques.* MIT Press, 2009.

[14] D. Koutra, T.-Y. Ke, U. Kang, D. H. Chau, H.-K. K. Pao, and C. Faloutsos. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *ECML/PKDD (2)*, 2011.

[15] Q. Lu and L. Getoor. Link-based classification. In *Machine Learning, ICML*, pages 496–503, 2003.

[16] S. A. Macskassy and F. J. Provost. Classification in networked data: A toolkit and a univariate case study. *JMLR*, 8:935–983, 2007.

[17] J. Mahmud, J. Nichols, and C. Drews. Where is this tweet from? inferring home locations of twitter users. In *Proc. ICWSM*, 2012.

[18] L. K. McDowell, K. M. Gupta, and D. W. Aha. Cautious collective classification. *JMLR*, 10:2777–2836, 2009.

[19] A. Mislove, B. Viswanath, P. K. Gummadi, and P. Druschel. You are who you know: inferring user profiles in online social networks. In *WSDM*, 2010.

[20] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8, 2007.

[21] J. Neville and F. Provost. Predictive modeling with social networks. ICWSM, Tutorial, 2009.

[22] A. Sadilek, H. Kautz, and J. P. Bigham. Finding your friends and following them to where you are. In *WSDM*, 2012.

[23] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient structured prediction with latent variables for general graphical models. In *ICML*, 2012.

[24] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[25] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *NIPS*, 2001.

[26] P. P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. Pereira. Weakly-supervised acquisition of labeled class instances using graph random walks. In *EMNLP*, 2008.

[27] B. Taskar, E. Segal, and D. Koller. Probabilistic classification and clustering in relational data. In *IJCAI*, 2001.

[28] U. Weinsberg, S. Bhagat, S. Ioannidis, and N. Taft. Blurme: inferring and obfuscating user gender based on ratings. In *RecSys*, 2012.

[29] R. Xiang and J. Neville. Pseudolikelihood em for within-network relational learning. In *ICDM*, 2008.

[30] R. Xiang and J. Neville. Collective inference for network data with copula latent markov networks. In *WSDM*, 2013.

[31] E. Zheleva and L. Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proc. WWW*, 2009.

[32] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2003.